REMARKS

The foregoing Amendment After Final and the following Remarks are submitted in response to the Final Office Action issued on December 1, 2005 in connection with the above-identified application, and are being filed within the three-month shortened statutory period set for a response to the Final Office Action.

Claims 1, 3, 7, 10, 11, 14-20, and 27-31 remain pending in the present application as amended. Independent claims 1 and 20 have been amended to more particularly point out and distinctly recite the subject matter that Applicants regard as their invention. Applicants respectfully submit that no new matter has been added to the application by the amendment.

The Examiner has rejected claims 1, 3, 6-11, 14, 15, 18-20, and 24-29 under 35 U.S.C. § 103(a) as being obvious over You (U.S. Patent No. 6,158,045) in view of Niemi et al. (U.S. Patent No. 6,470,388). Applicants respectfully traverse the You-Niemi rejection insofar as it may be applied to the claims as amended.

Independent claim 1 as amended recites a debugger for debugging any of a plurality of debuggees. Each debuggee has a debugging type attribute selected from a plurality of debugging type attributes and representative of a type of debugging to be performed with respect to the debuggee, and each debuggee also has a processor attribute selected from a plurality of processor attributes and representative of a type of processor associated with the debuggee. The debugger is instantiated on a computer and has a single debugger engine for performing debugging functions with respect to any of the plurality of debuggees.

The engine includes a plurality of debugging type blocks, where each debugging type block supports at least one of the plurality of debugging type attributes, and a plurality of processor blocks, where each processor block supports at least one of the plurality of processor attributes. A particular debugging type block and a particular processor block are selected for debugging a particular debuggee based on the debugging type attribute and processor attribute of the particular debuggee.

Significantly, the plurality of debugging type blocks are organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type. The debugging type abstraction comprises programming code, and at least a portion of the programming code for the debugging type abstraction is common as between at least some debugging type blocks and is shared by such debugging type blocks. In particular, the programming code for the debugging type abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom. The debugging type abstraction nodes include leaf nodes from which no other nodes branch out. Significantly, each debugging type block is defined to include a plurality of nodes extending from the base node to a particular leaf node.

Likewise, the plurality of processor blocks are organized into a processor abstraction available to provide processor services that vary in implementation for each processor. As with the debugging type abstraction, the processor abstraction comprises programming code, and at least a portion of the programming code for the processor abstraction is common as between at least some processor blocks and is shared by such processor blocks. Also as with the debugging type abstraction, the programming code for the

processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom. As with the debugging type abstraction, the processor abstraction nodes include leaf nodes from which no other nodes branch out. Significantly, each processor block is defined to include a plurality of nodes extending from the base node to a particular leaf node.

Independent claim 20 recites subject matter similar to that of claim 1, albeit in the form of a computer with the debugger instantiated thereon.

As was previously pointed out, the You reference discloses a debugger portable to multiple operating systems and hardware platforms. However, Applicants respectfully submit that the You reference does not disclose or suggest a debugger with:

> - a plurality of debugging type blocks organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type and a plurality of processor blocks organized into a processor abstraction available to provide processor services that vary in implementation for each processor,
> - at least a portion of the programming code for the debugging type abstraction is common as between at least some debugging type blocks and is shared by such debugging type blocks and at least a portion of the programming code for the processor abstraction is common as between at least some processor blocks and is shared by such processor blocks, and
> - the programming code for the debugging type abstraction organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each debugging type block is defined to include a plurality of nodes extending from the base node to a particular leaf node, and the programming code for the processor abstraction likewise organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each processor block is defined to include a plurality of nodes extending from the base node to a particular leaf node,

all as is additionally required by claims 1 and 20. Particularly with regard to the "plurality of nodes extending from the base node to a particular leaf node" language, Applicants respectfully note that the Examiner points out in a Response to Arguments section of the

Final Office Action that such language was previously argued but not recited in the claims. Accordingly, in amending the claims, Applicants have in fact recited such language.

Applicants respectfully note that the Examiner again points to Fig. 9 of the You reference as showing such a tree form. However, and again, Applicants respectfully point out that such Fig. 9 shows an addressing abstraction utilized to facilitate the use of target memory addresses in a portable fashion (Abstract), and that only a particular one of the nodes within such addressing abstraction is selected to locate a particular address, depending on operating system and/or platform. In contrast, the present invention as recited in claims 1 and 20 requires that the programming code for both the debugging type abstraction and the processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each respective block is defined to include a *plurality* of nodes *extending from the base node to a particular leaf node.*

Put another way, the You reference and the addressing abstraction tree of Fig. 9 thereof do not appreciate that using such tree is not merely a matter of selecting a particular node thereof, but selecting a plurality of such nodes extending from a base node to a particular leaf node to form a particular debugging type block or processor block, as the case may be. That is, the tree of Fig. 9 has a plurality of nodes, each for a particular type of address, where selecting a particular node is necessary for selecting the corresponding type of address. In contrast, the tree recited in claims 1 and 20 has a plurality of nodes where a particular debugging type block or processor block is defined by selecting a plurality of such nodes extending from a base node to a particular leaf node, as may best be appreciated with reference to Figs. 4 and 5 of the present application.

While the Niemi reference may be said to disclose a debugging type abstraction, such Niemi reference likewise fails to disclose or suggest that such a debugging type abstraction is employed and structured in the manner recited in claims 1 and 20, as set forth above, and that a corresponding processor abstraction likewise be employed and structured in the manner recited in claims 1 and 20, as set forth above.

Thus, the combination of the You and Niemi references does not result in the subject matter recited in claims 1 and 20. Accordingly, Applicants respectfully submit that the You and Niemi references cannot be applied to make obvious claims 1 or 20 or any claims depending therefrom. As a result, Applicants respectfully request reconsideration and withdrawal of the You-Niemi rejection.

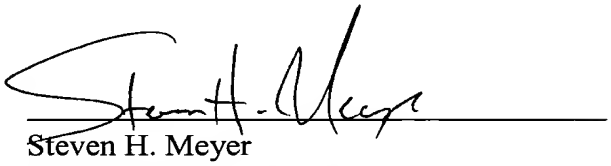
Claims 16, 17, 30, and 31 stand rejected under § 103(a) as being obvious over the You and Niemi references and further in view of Hawley et al. (U.S. Patent No. 5,533,192). Applicants respectfully traverse the You-Niemi-Hawley rejection insofar as it may be applied to the claims as amended.

Applicants respectfully point out that since independent claims 1 and 20 are unanticipated and have been shown to be non-obvious, then so too must all claims depending therefrom including claims 16-17 and 30-31 be unanticipated and non-obvious, at least by their dependency. As a result, Applicants respectfully request reconsideration and withdrawal of the You-Niemi-Hawley rejection.

In view of the foregoing, Applicants respectfully submit that the present application including claims 1, 3, 7, 10, 11, 14-20, and 27-31 is in condition for allowance, and such action is respectfully requested.

Respectfully submitted,

Date: February 10, 2005



Steven H. Meyer
Registration No. 37,189

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439